6.895 PCP and Hardness of Approximation — MIT, Fall 2010

# Lecture 3: Coding Theory

Lecturer: *Dana Moshkovitz* — Scribe: *Michael Forbes and Dana Moshkovitz*

## 1 Motivation

In the course we will make heavy use of coding theory. In this lecture we review the basic ideas and constructions. We start with two motivating examples.

### 1.1 Noisy Communication

Alice gets a $k$-bit message $x$ to send to Bob. Unfortunately, the communication channel is corrupted, and some number of bits may flip. For conceretness, say twenty percent of all bits sent on the channel may flip. Which bits are flipped is unknown, and different bits may flip for different messages sent on the channel. Can Alice still send a message that can be reliably received by Bob? We allow Alice and Bob to make any agreements prior to Alice learning about her message $x$. We also allow Alice to send more than $k$ bits on the channel. The redundant bits can compesate for the bits lost due to corruption.

We will see that Alice can send only $n = \Theta(k)$ bits of information about her message, such that even if twenty percent of those bits are flipped, the mistakes can be detected, and all of the original $k$ bits can be recovered by Bob!

### 1.2 Equality Testing

The following example is a classic example from communication complexity. Its setup and ideas are very useful for understanding PCPs.

Suppose there are two players, Alice and Bob, as well as a verifier. Each player receives a private $k$-bit string. Alice's string is denoted $x$, and Bob's string is denoted $y$. The verifier interacts with Alice and Bob to answer the question "Is $x = y$?". The verifier sends each of Alice and Bob a question about their strings, and receives back an answer. Given the two answers, the verifier either declares "equal" or declares "unequal". We allow all parties to make any arrnagements prior to receiving the strings. The goal is to minimize the number of bits transmitted by the verifier and by the players after receiving the strings.

A possible solution is for the verifier to ask each of the players for her/his entire string. This gives a protocol with $\Theta(k)$ bits of communication. This can be shown to be optimal for a deterministic verifier. A randomized verifier, however, can achieve much lower communication complexity. The randomized setting is as follows. the verifier has private randomness $r$, and it sends questions to Alice and Bob that depend on $r$. The protocol should satisfy:

- **Completeness**: $x = y \implies \Pr_r[\text{verifier declares equal}] = 1$
- **Soundness**: $x \neq y \implies \Pr_r[\text{verifier declares equal}] \leq 0.9$

Note that the 0.9 in the soundness is high, but bounded away from 1. It can be improved to an arbitrarily small constant by repeating the protocol a sufficient number of times. This increases the communication complexity only by a constant factor.

We will see how to provide a randomized protocol where the verifier sends $\Theta(\log k)$ bits, and Alice and Bob each replies with just one bit!

### 1.3 Solution to Problems

Both of the above problems can be solved using *coding theory.*

Let us make some relevant definitions. The space of strings over an alphabet $\Sigma$ has a natural distance metric:

**Definition 1.** *For strings $x, y \in \Sigma^n$, the **Hamming distance** is denoted $\Delta(x,y) = \frac{|\{i|x_i \neq y_i\}|}{n}$.*

We have the following easy lemma.

**Lemma 1.1.** *$\Delta$ is a metric. That is, it satisfies for every $x, y \in \Sigma^n$:*

- *$\Delta(x,y) \geq 0$; $\Delta(x,y) = 0 \iff x = y$.*
- *Symmetry: $\Delta(x,y) = \Delta(y,x)$*
- *Triangle Inequality: $\Delta(x,y) \leq \Delta(x,z) + \Delta(z,y)$*

Let us say a transformation $E : \Sigma^k \to \Sigma^n$ is *$c$-distancing* if for every two different strings $x \neq y \in \Sigma^n$, the transformation maps $x$ and $y$ to points of distance at least $c$, i.e., $\Delta(E(x), E(y)) \geq c$. Thus, no matter how close $x$ and $y$ were originally (as long as $x \neq y$), they get mapped to vastly different strings under the map. Coding theory deals with the understanding and the construction of distancing transformations. It turns out that $c$-distancing transformations exist for constant $c$ (depending on $\Sigma$) and $n = \Theta(k)$. In particular, one can take $c > 2/5$ and $\Sigma = \{0,1\}$.

Appropriate distancing tranformation solves the two motivating problems we presented. For the noisy communication example, Alice and Bob agree on a distancing transformation $E$ ahead of time. Alice then sends $E(x)$ on the noisy channel to Bob. Bob receives a version $w$ of $E(x)$ with at most twenty percent of the bits flipped, so $\Delta(w, E(x)) \leq \frac{1}{5}$. Now Bob recovers $x$ by finding the closest $E(\cdot)$ to $w$. If $c > \frac{2}{5}$, Bob will necessarily recover $x$. Why? If Bob recovered $x' \neq x$ then $\Delta(w, E(x')) \leq \frac{1}{5}$, and $\Delta(E(x'), E(x)) \leq \frac{2}{5}$.

For the equality testing example, all parties agree on a distancing transformation $E$ to begin with. Then, the verifier picks at random $i \in [n]$. The verifier asks Alice about $E(x)_i$, and it asks Bob about $E(y)_i$. It declares the strings are equal if $E(x)_i = E(y)_i$.

If the strings are equal the verifier always declares equality. On the other hand, if the strings are not equal, by the property of the distancing transformation, the two bits are not equal with probability at least $c$. If $c \geq \frac{1}{10}$, the verifier declares quality with probability at most 0.9.

The verifier sends $i$ to both players, and this requires $\Theta(\log n)$ bits. If $n = \Theta(k)$, then $\log n = \Theta(\log k)$. Each of the players replies with just one bit!

With these answers in mind, we now give the formal definitions of coding theory.

## 2 Coding Theory

The basic definition of coding theory is that of a code:

**Definition 2.** *A **code** is a subset $C \subseteq \Sigma^n$. It is called an $(n,k,d)_\Sigma$ code if:*

- $|C| = |\Sigma|^k$

- *For all $x \neq y \in C$, $\Delta(x,y) \geq d/n$*

*We call*

- *$k$ the dimension*
- *$n$ the length*
- *$d$ the distance*
- *$\Sigma$ the alphabet, when omitted we assume $\Sigma = \{0,1\}$*
- *$d/n$ the relative distance (often just called "distance")*
- *$k/n$ the rate*

*If $E : \Sigma^k \to \Sigma^n$ is such that $E(\Sigma^k) = C$, then we call $E$ an encoding function of $C$. An encoding function maps every possible "message" $x \in \Sigma^k$ to a codeword of $C$.*

When we construct codes we want to maximize the relative distance as well as the rate, and we want to minimize the alphabet size.

## 2.1 Tradeoffs Between Code Parameters

There are various tradeoffs between code parameters. Next we review some of them.

The Singleton bound says that the sum of the relative distance and the rate is, up to an additive $\frac{1}{n}$, at most 1. Thus, if the relative distance is close to 1, the rate must be close to 0, and vice versa:

**Lemma 2.1** (Singleton Bound)**.** *For any $(n,k,d)_\Sigma$ code, $n \geq k + d - 1$.*

*Proof.* Consider the first $k-1$ coordinates of the codewords. As there are $|\Sigma|^k$ codewords, there must be distinct codewords $x \neq y \in C \subseteq \Sigma^n$, such that $x$ and $y$ agree on the first $k-1$ coordinates. However, as $\Delta(x,y) \geq d$, it must be that there are at least $d$ coordinates in which $x$ and $y$ disagree. Thus, there must be at least $k-1+d$ coordinates. As $n$ is the number of coordinates, the claim follows. $\square$

Any code that meets the singleton bound is called a *maximal distance separable* (MDS) code. Later in the lecture we will see such a code.

The Plotkin bound states that the relative distance is at most $\approx 1 - \frac{1}{|\Sigma|}$. Hence, for the reltative distance to be close to 1, the alphabet must be sufficiently large:

**Lemma 2.2** (Plotkin Bound)**.** *For any $(n,k,d)_\Sigma$ code $C$,*

$$\frac{d}{n} \leq \frac{|C|}{|C|-1}\left(1 - \frac{1}{|\Sigma|}\right).$$

*Proof.* Observe the sum:

$$\sum_{x \in C} \sum_{y \in C} \Delta(x,y). \tag{1}$$

By the code distance property, we have a lower bound of $|C|\left(|C|-1\right)\frac{d}{n}$ on the sum.

For every $i \in [n]$, and every $\sigma \in \Sigma$, let $x_{i,\sigma}$ be the number of codewords that have $\sigma$ in their $i$'th coordinate. So, for every $i \in [n]$, $\sum_{\sigma\in\Sigma} x_{i,\sigma} = |C|$. Then, every $i$ contributs $\frac{1}{n}\sum_{\sigma\neq\sigma'\in\Sigma} x_{i,\sigma}x_{i,\sigma'}$ to the sum.

$$\sum_{\sigma\neq\sigma'\in\Sigma} x_{i,\sigma}x_{i,\sigma'} = \left(\sum_{\sigma\in\Sigma} x_{i,\sigma}\right)^2 - \sum_{\sigma\in\Sigma} x_{i,\sigma}^2 = |C|^2 - \sum_{\sigma\in\Sigma} x_{i,\sigma}^2.$$

By Jensen's inequality, stating that for a convex function $f : \Re \to \Re$ (here: $f(x) = x^2$), we have $\mathbf{E}[f(x)] \geq f(\mathbf{E}[x])$,

$$\frac{1}{|\Sigma|}\sum_{\sigma\in\Sigma} x_{i,\sigma}^2 \geq \left(\frac{1}{|\Sigma|}\sum_{\sigma\in\Sigma} x_{i,\sigma}\right)^2 = \frac{|C|^2}{|\Sigma|^2}.$$

Hence,

$$\sum_{\sigma\neq\sigma'\in\Sigma} x_{i,\sigma}x_{i,\sigma'} \leq |C|^2\left(1 - \frac{1}{|\Sigma|}\right).$$

By combining this upper bound with the lower bound on the sum (1), we get:

$$\frac{d}{n} \leq \frac{|C|}{|C|-1}\left(1 - \frac{1}{|\Sigma|}\right).$$

□

## 2.2 Linear Codes

An important family of codes is that of linear codes. Their extra structure allows for succinct representation and further applications.

**Definition 3.** *A **linear code** $C$ with parameters $[n,k,d]_\Sigma$ is a code where $\Sigma$ is a finite field, and $C$ is a linear subspace of $\Sigma^n$.*

Note that as linear codes are subspaces they can be succinctly described by a basis. The linear transformation associated with the encoding is given by a matrix referred to as a *generating matrix*. Linear codes are invariant under translation of a codeword, which leads to the next lemma.

**Lemma 2.3.** *The relative distance $d/n$ of a linear code $C$ is*

$$\min_{\vec{0}\neq x\in C} \Delta(x, \vec{0}).$$

*Proof.* Observe that for any three vectors over a finite field, $x$, $y$ and $z$, we have $\Delta(x,y) = \Delta(x+z, y+z)$. Taking these as words in a linear code, we see that $x+z$ and $y+z$ are also words in the code. Taking $z = -x$ (also in the code, by linearity) we see that $\Delta(x,y) = \Delta(0, y-x)$. Note that $x \neq y$ iff $y - x \neq \vec{0}$. □

We can also use row-reduction to transform a linear code into a normal form. We call an encoding $E : \Sigma^k \to \Sigma^n$ *systematic* if it can be expressed as $E(x) = x, E'(x)$, for some map $E' : \Sigma^k \to \Sigma^{n-k}$.

**Lemma 2.4.** *Any $[n,k,d]_\Sigma$ linear code has a systematic encoding.*

*Proof.* Observe that the encoding procedure in a linear code can be expressed as a matrix transformation. That is, the generating matrix $G$ is $n \times k$, and the encoding function is $Gx$. Notice that $G$ must have rank at least $k$, for otherwise the encoding procedure would not be injective, which was assumed by definition. Thus, we can row-reduce $G$ such that the first $k$ rows form a $k \times k$ identity matrix. The resulting matrix $G'$ is a systematic code. □

For a linear code $C \subseteq \Sigma^n$, we define its *dual code* to be $C^\perp$, where $C^\perp = \{w \in \Sigma^n | \forall v \in C, \langle w, v\rangle = 0\}$ and $\langle .,. \rangle : \mathbb{F}^n \times \mathbb{F}^n \to \mathbb{F}$ is the inner product with respect to the standard basis. Observe that $C^\perp$ is also a linear subspace. The elements of $C^\perp$ correspond to the linear equations that elements of $C$ satisfy. A matrix whose rows are a basis to $C^\perp$ is called a *parity check matrix* of $C$. The following lemma tells us that $C^\perp$ also determines $C$:

**Lemma 2.5.** *Let $V$ be a subspace of $\mathbb{F}^n$. Then $(V^\perp)^\perp = V$.*

*Proof.* Observe that $V \subseteq (V^\perp)^\perp$ by definition (and this holds in any inner-product space). To show the equality, we show that they have the same dimension. This can be done by taking a basis $(v_1, \ldots, v_k)$ of $V$ and considering the map $\varphi : \mathbb{F}^n \to \mathbb{F}^k$ defined by $w \mapsto \langle v_1, w\rangle, \ldots, \langle v_k, w\rangle$. By the Rank-Nullity theorem, $\dim Im\varphi + \dim Ker\varphi = n$. But $\dim Im\varphi = k$ because we chose a basis of $V$. So as the kernel of this map is $V^\perp$, we see that $\dim V + \dim(V^\perp) = n$, and so $\dim V = \dim(V^\perp)^\perp$. As $V \subseteq (V^\perp)^\perp$, this gives the equality. □

## 3 Constructions of Codes

### 3.1 Random Codes

Consider the linear code $C$ defined by a random generating matrix, i.e., a matrix that each of its entries is drawn independently uniformly at random from $\Sigma$. This results in, with high probability, a code with constant rate and constant relative distance:

**Theorem 4** (Gilbert-Varshamov). *For any $0 < \delta < 1 - \frac{1}{|\Sigma|}$ there is $\alpha > 0$, such that for sufficiently large $n$ and $k = \alpha n$, the code $C$ is a $[n, k, \delta n]_\Sigma$ code with high probability.*

That is, a random linear code is extremely likely to be an excellent code, or, put differently, most linear codes are excellent codes. Still, finding an explicit, concrete linear code that is known to have good parameters is a very challenging task. In the sequel we survey some basic constructions of codes; all be useful later. We eventually show how to find efficiently codes with good parameters.

### 3.2 The Long Code

For a $(n, k, d)_\Sigma$ code, an encoding function can be thought of as $n$ functions $\Sigma^k \to \Sigma$, one for each coordinate in the output. The **long code** is obtained by taking *all* possible functions $\Sigma^k \to \Sigma$. There are $|\Sigma|^{|\Sigma|^k}$ such functions, so $n = |\Sigma|^{|\Sigma|^k}$, and we associate each coordinate in $\{1, \ldots, n\}$ with a function $f : \Sigma^k \to \Sigma$. A message $x \in \Sigma^k$ is encoded by a codeword whose $f$'th position is $f(x)$. The rate of the long code is very poor $k/|\Sigma|^{|\Sigma|^k}$.

On the other hand, the relative distance is $1 - 1/|\Sigma|$. This is optimal by Lemma 2.2. To analyze the distance, observe that for $x \in \Sigma^k$ and for every $\sigma \in \Sigma$, a random function $f : \Sigma^k \to \Sigma$ will have $f(x) = \sigma$ with probability $\frac{1}{|\Sigma|}$. In particular, for $y \neq x \in \Sigma^k$, the probability that $f(x) = f(y)$ is $\frac{1}{|\Sigma|}$.

The long code is clearly a non-linear code even when $\Sigma$ is a finite field, because some of the functions $\Sigma^k \to \Sigma$ are non-linear. Still, the unique structure of the long code will prove to be useful for our purposes. It will be used when $\Sigma^k$ is constant.

## 3.3 The Hadamard Code

The long code evaluates all distinct functions on the input. The **Hadamard Code** evaluates only (homogeneous) linear functions on the input. That is, over a finite field $\mathbb{F}$, the Hadamard code encodes a vector $x \in \mathbb{F}^k$ as $(\langle a, x\rangle)_{a\in\mathbb{F}^k}$. This makes the Hadamard code a linear code. One can see that the length is $n = |\mathbb{F}|^k$. This corresponds to rate $k/|\mathbb{F}|^k$, which is much better than that of the long code, but is still quite poor. Thus, the Hadamard code is useful mainly in application where $|\mathbb{F}|^k$ is sufficiently small.

The distance of the Hadamard code is still optimal $1 - 1/|\mathbb{F}|$: Given $x \neq y$, find an index $i$ where $x_i \neq y_i$. Then, if we condition on all of the indices in $a$ that are not $i$, the probability that $\langle a, x\rangle = \langle a, y\rangle$ boils down to this last unconditioned index, which is distributed evenly. Thus, the probability of equality is $1/|\mathbb{F}|$.

An alternative view of the Hadamrd encoding is that it interperts its message $x \in \mathbb{F}^k$ as defining a (homogeneous) linear function $l_x : \mathbb{F}^k \to \mathbb{F}^n$ with $l_x(a) = \langle a, x\rangle$. The encoding is then an evaluation of the function $l_x$ on all points $a \in \mathbb{F}^k$. This viewpoint is very useful, and also leads naturally into other codes.

## 3.4 Reed-Solomon (RS) Codes

Reed-Solomon codes generalize Hadamard codes in a natural way. Instead of treating the input as a *(homogeneous) linear form*, we can consider the input $c \in \mathbb{F}^k$ as representing a univariate polynomial of bounded degree, $p_c : \mathbb{F} \to \mathbb{F}$, where $p_c(t) = \sum_{i=0}^{k-1} c_i t^i$. A codeword will be the evaluation of $p_c$ on all points $t \in \mathbb{F}$. This yields $n = |\mathbb{F}|$, so that the rate is $k/|\mathbb{F}|$. If $k = \Theta(|\mathbb{F}|)$, this finally yields constant rate!

Further, we can see that Reed-Solomon is a linear code, as by definition, for any $c, c' \in \mathbb{F}^k$, for any $\alpha, \beta, t \in \mathbb{F}$ we have: $\alpha p_c(t) + \beta p_{c'}(t) = (\alpha p_c + \beta p_{c'})(t)$.

Thus, to examine the distance we only need to look at the distance of a codeword from the zero codeword. Recalling that a non-zero univariate polynomial of degree at most $k-1$ has at most $k-1$ roots, we see that the relative distance of the code is least $1 - (k-1)/|\mathbb{F}|$.

Reed-Solomon codes are MDS codes — that is, they meet the singleton bound and have an optimal relative distance to rate tradeoff (see Lemma 2.1). Reed Solomon codes are also extremely useful in practice, and are even used in CD-ROM's and later technologies. The big disadvantage of Reed-Solomon codes is that their alphabet size is as large as their length.

## 3.5 Reed-Muller Codes

The **Reed-Muller Codes** generalize the Reed-Solomon codes to the multivariate case. At the same time they generalize the Hadamard code to larger degrees.

The input to the encoding is interpreted as a polynomial in $m$ variables of degree at most $r$. The output is the evaluation of this polynomial on all points in the space $\mathbb{F}^m$, so the length is $n = \mathbb{F}^m$.

To analyze the distance, we first need the Schwartz-Zippel lemma:

**Theorem 5** (Schwartz-Zippel Lemma). *Let $f \in \mathbb{F}[x_1, \ldots, x_m]$ be an $m$-variate polynomial of degree at most $r$ that is not identically zero. Then*

$$\Pr_{x \in \mathbb{F}^m}[f(x) = 0] \leq r/|\mathbb{F}|$$

*Proof.* Observe that for $m = 1$ this is the usual bound on the number of roots of a univariate polynomial. We assume $r < |\mathbb{F}|$ as otherwise the result is trivial. For $m > 1$ we reduce to the $m = 1$ case. First we need the following lemma.

**Lemma 3.1.** *Let $f \in \mathbb{F}[x_1, \ldots, x_m]$ be an $m$-variate polynomial of degree at most $r < |\mathbb{F}|$ that is not identically zero. Then there exists $y \in \mathbb{F}^m$, where $f(y) \neq 0$.*

*Proof.* We do via contradiction, so assume $f(y) = 0$ for all $y$. Consider all ways to assign $x_1, \ldots, x_i$ to elements in $\mathbb{F}$, for all $0 \leq i \leq n$. Call these *partial assignments*. The partial assignments naturally form a tree. Further, each partial assignment induces a polynomial on the rest of the variables. Clearly, any assignment that assigns all of the variables induces the zero element upon assignment, as per assumption. Further, the assignment that assigns none of the variables (that is, $i = 0$) leaves the polynomial unchanged.

Thus, any traversal down the tree necessary reaches a partial assignment $a_1, \ldots, a_{i_0-1}$ that induces a non-zero polynomial, but where any further assignment (of $x_{i_0}$) induces the zero polynomial in the rest of the variables. One can think of this as having a polynomial $g$ in the field $\mathbb{F}(x_{i_0+1}, \ldots, x_m)$ (the field of rational functions in the variables $x_{i_0+1}, \ldots, x_m$) in the variable $x_{i_0}$. Further, this polynomial is non-zero, but evaluates to zero at $|\mathbb{F}|$ points — one point for each possible assignment of $x_{i_0}$. Notice that $g$ was induced from $f$, so $\deg g \leq \deg f \leq r$. But this is a contradiction as a non-zero, univariate polynomial of degree at most $r < |\mathbb{F}|$ cannot evaluate to zero on $|\mathbb{F}|$ points. (End of Lemma 3.1) □

The above lemma shows that $f$ must be non-zero somewhere, and we now bootstrap this to show that $f$ must be non-zero on a noticeable fraction of the points. Assume without loss of generality that $f$ is of degree exactly $r$. Thus, we can decompose $f$ into $f = f_{=r} + f_{<r}$, where $f_{=r}$ is a degree $r$ homogeneous polynomial that is not identically zero, and $\deg f_{<r} < r$. By the above lemma, there is some point $y$ where $f_{=r}(y) \neq 0$. As $f_{=r}$ is homogeneous, $f(\vec{0}) = 0$, so $y \neq 0$.

Therefore, we can now partition the space $\mathbb{F}^m$ into the lines $l_x = \{x + ty | t \in \mathbb{F}\}$ with direction $y$. This partition is possible as $y \neq 0$. Observe that there are $|\mathbb{F}^{m-1}|$ lines. Further, when we restrict $f$ to a line $l_x$, we see that $f(x+ty) = f_{=r}(x+ty) + f_{<r}(x+ty)$ is a degree $r$ polynomial, where the coefficient of $t^r$ is exactly $f_{=r}(y) \neq 0$. Thus, $f|_{l_x}$ is a non-zero univariate polynomial of degree exactly $r$, and hence has at most $r$ roots. So on each line, $f$ evaluates at most $r/|\mathbb{F}|$ fraction of the line to zero, and overall evaluates at most $r/|\mathbb{F}|$ fraction of $\mathbb{F}^m$ to 0. □

**Remark 3.1.** *The Schwartz-Zippel lemma is tight, with respect to its probability. For, we can consider any univariate polynomial with $r$ distinct roots.*

We can now return to Reed-Muller codes. Just as with Reed-Solomon codes, Reed-Muller codes are linear. Thus, analyzing the distance of the code amounts to analyzing the distance of a codeword from the zero codeword. The Schwartz-Zippel Lemma gives us that the distance is $1 - r/|\mathbb{F}|$, just like the distance of Reed-Solomon codes. There are $\binom{m+r}{m}$ monomials of degree at most $r$, so the dimension is $k = \binom{m+r}{m}$, and thus the rate is $\binom{m+r}{m}/|\mathbb{F}^m|$. For $m = 1$ we get the Reed-Solomon code. When $m > 1$, yet $m = \Theta(1)$, and $r = \Theta(|\mathbb{F}|)$, the rate is constant, and the alphabet $\mathbb{F}$ is much smaller than the length. This remedies the disadvantage of Reed-Solomon code, at the expense of a somewhat worse rate.

## 4 Operations on Codes

We saw that Reed-Solomon codes meet the Singleton bound, i.e., have an optimal relative distance to rate tradeoff. However, there was a drawback because Reed-Solomon codes require a large alphabet. One might hope that there is a way to modify Reed-Solomon codes into binary codes while retaining their relative distance and rate. In general, one can ask: can we take codes that some of their parameters are desirable and some are lacking, and transform them into codes where all parameters are desirable? will see two operations and examine their properties.

### 4.1 Concatenation of Codes

This operation is motivated by the question of how to reduce the alphabet size. While this operation is called "concatenation", it can also be thought of as *composition* of two codes: the first code has large alphabet, and the second code has small alphabet and small dimension. Concatenation replaces each alphabet symbol of a codeword of the first code with its encoding using the second code.

**Definition 6.** *Let $A$ be a code with parameters $(N, K, D)_{\Sigma^k}$ and encoding function $E$, and $B$ be a code with parameters $(n, k, d)_\Sigma$ and encoding function $F$. Define $\varphi : (\Sigma^k)^K \to \Sigma^{nN}$ by*

$$E(x_1, \ldots, x_K) = (e_1, \ldots, e_N) \Rightarrow \varphi(x_1, \ldots, x_K) = (F(e_1), \ldots, F(e_N)).$$

*Define $E \diamond F : \Sigma^{kK} \to \Sigma^{nN}$ to be the encoding function derived from $\varphi$ via the identification of $(\Sigma^k)^K$ with $\Sigma^{kK}$. Call the resulting code $A \diamond B$, the **concatenation** of $A$ and $B$. Call $A$ the **outer code**, and call $B$ the **inner code**.*

**Lemma 4.1.** *Let $A$ be a code with parameters $(N, K, D)_{\Sigma^k}$ and encoding function $E$, and $B$ be a code with parameters $(n, k, d)_\Sigma$ and encoding function $F$. Then $A \diamond B$ is a $(nN, kK, dD)_\Sigma$ code. Moreover, if $A$ and $B$ (as well as $E$ and $F$) are linear, then so is $A \diamond B$.*

*Proof.* The dimension, length, and alphabet parameters of the concatenated code, as well as linearity, are clear. We now examine the distance. Consider $x \neq y \in (\Sigma^k)^K$, which we use instead of $\Sigma^{kK}$. Then observe that via the properties of $A$, at least $D/N$ fraction of symbols (in $\Sigma^k$) in $E(x)$ differ from $E(y)$. Thus, when we apply the encoding $F$ to each of these differing symbols (thought now of as sequences of $k$ symbols in $\Sigma$), we get a fraction of $d/n$ differing symbols over $\Sigma$. Thus, we see that overall there are at at least $dD/(nN)$ fraction of differing symbols. □

As an example, we consider RS$\diamond$Hadamard, where we have a Reed-Solomon outer code and a Hadamard inner code. Even though the Hadamard code does not have a good rate, the contribution of its length to the total length is moderate, as it is only used for encoding of relatively short messages. Consider the Reed-Solomon code with parameters $[q^k, d+1, 1 - d/q^k]_{\mathbb{F}_{q^k}}$. The Hadamard code over $\mathbb{F}_q$ has parameters $[q^k, k, 1 - 1/q]_{\mathbb{F}_q}$. Thus, by the above, the concatenation has parameters $[q^{2k}, (d+1)k, (1 - d/q^k)(1 - 1/q)]_{\mathbb{F}_q}$. In particular, if we take

$d = \epsilon q^k$ for $0 < \epsilon < 1$, we get a code with parameters $[q^{2k}, \epsilon k q^k, 1 - 1/q - \epsilon]_{\mathbb{F}_q}$. This gives a roughly-quadratic length encoding with a near optimal relative distance over small alphabets.

Using code concatenation, we can find in polynomial time in the length $n$ codes with constant rate and constant relative distance over any constant alphabet $\Sigma$:

**Theorem 7.** *For any constant alphabet $\Sigma$, for any $0 < \delta < 1 - \frac{1}{|\Sigma|}$, there is $\alpha > 0$, such that for sufficiently large $n$, one can find an explicit $[n, \alpha n, \delta n]$-code in time polynomial in $n$.*

*Proof.* Let $\delta = \delta_1 \cdot \delta_2$ where $\delta_2 < 1 - \frac{1}{|\Sigma|}$. Reed-Solomon code is a $[n_1, k_1 = (1-\delta_1)n_1, \delta_1 n_1]_{\{1,\dots,n_1\}}$ code.

Let $k_2 = \log n_1 / \log |\Sigma|$. Obtain a $[n_2, k_2, \delta_2 n_2]_\Sigma$ code with $n_2 = \Theta(k_2)$ and $\delta_2 = \Theta(1)$ by a brute-force search. Note that: (1) By Theorem 4, there is such a code; (2) The time complexity of the search is $\Theta(|\Sigma|^{n_2}) = \Theta(|\Sigma|^{\Theta(k_2)}) = n_1^{\Theta(1)}$.

Concatenate the two codes to obtain a $[n_1 n_2, k_1 k_2, \delta_1 \delta_2 n_1 n_2]_\Sigma$ code. Note that the rate $k_1 k_2 / n_1 n_2$ is constant, and the relative distance is $\delta$. □

## 4.2 Tensor Product

In this section we see an operation whose purpose is to increase the dimension of a code. Suppose we have a code with desireable tradeoffs between code parameters, but with very few codewords. Such could be obtained by an expensive brute-force search among all possible codes. We want to obtain a code with similar tradeoffs, but higher dimension. The operation we show is called *tensor product* or *direct product* of two codes. The new encoding interprets the message as a matrix, uses the first code to encode the rows of the matrix, and then uses the second code to encode the resulting columns. If, for instance, both codes had dimension $k$, the new code has the much higher dimension $k^2$.

**Definition 8.** *For $i \in \{1,2\}$, suppose $C_i$ is a code with parameters $[n_i, k_i, d_i]_\Sigma$ and a linear encoding function $E_i : \Sigma^{k_i} \to \Sigma^{n_i}$. Consider the following encoding function $E_1 \otimes E_2 : \Sigma^{k_1 \times k_2} \to \Sigma^{n_1 \times n_2}$, defined by the map:*

$$\begin{bmatrix} x_{1,1} & \cdots & x_{1,k_2} \\ \vdots & \ddots & \vdots \\ x_{k_1,1} & \cdots & x_{k_1,k_2} \end{bmatrix} \mapsto \begin{bmatrix} y_{1,1} & \cdots & \cdots & y_{1,n_2} \\ \vdots & \ddots & \ddots & \vdots \\ y_{k_1,1} & \cdots & \cdots & y_{k_1,n_2} \end{bmatrix} \mapsto \begin{bmatrix} z_{1,1} & \cdots & \cdots & z_{1,n_2} \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ z_{n_1,1} & \cdots & \cdots & z_{n_1,n_2} \end{bmatrix}$$

*where $E_1(x_{i,1}, \dots, x_{i,k_2}) = (y_{i,1}, \dots, y_{i,n_2})$, and $E_2(y_{1,i}, \dots, y_{k_1,i}) = (z_{1,i}, \dots, z_{n_1,i})$.*

*Call the code that this encoding function induces the* ***tensor product*** *of $C_1$ and $C_2$, denoted $C_1 \otimes C_2$.*

*If we tensor a code with itself $m$ times, we denote this $C^{\otimes m}$.*

**Lemma 4.2.** *For $i \in \{1,2\}$, suppose $C_i$ is a code with parameters $(n_i, k_i, d_i)_\Sigma$ and encoding function $E_i : \Sigma^{k_i} \to \Sigma^{n_i}$. Then $C_1 \otimes C_2$ is a $(n_1 n_2, k_1 k_2, d_1 d_2)_\Sigma$ code.*

*Proof.* All of the parameters are clear except for the distance parameter. Suppose we have two $k_1 \times k_2$ matrices over $\Sigma$. Call the matrices formed from $X$ in the encoding process of $C_1 \times C_2$ to be $X'$ and $X''$, and similarly for $Y$. If $X \neq Y$, then there is some $i$ such that row $i$ in $X$ and $Y$ differ. Thus, the row $i$ in $X'$ and $Y'$ must have at least $d_1$ differences by properties of $C_1$. Thus, we then see that $X'$ and $Y'$ have at least $d_1$ columns in which they differ. Thus, we see that in $X''$ and $Y''$, there are at least $d_1$ columns that differ by $d_2$ coordinates. Thus, there are at least $d_1 d_2$ differences total. □

The operation of a tensor product is a linear algebra operation. Given $y \in \mathbb{F}^n$ and $z \in \mathbb{F}^m$, their tensor product is the vector $y \otimes z \in \mathbb{F}^{n \times m}$ where the $(i,j)$-th coordinate has value $y_i z_j$. The vector $y \otimes z$ is conveiently interpreted as an $n \times m$ matrix over $\mathbb{F}$. All its rows are multiples of $z$, and all the columns are multiples of $y$.

The tensor product of $C_1$ and $C_2$ consists of the matrices with rows from $C_1$ and columns from $C_2$:

**Lemma 4.3.** $C_1 \otimes C_2 = \mathit{span}\,\{y \otimes z \mid y \in C_1, z \in C_2\}$.

*Proof.* Without loss of generality, $E_1$ and $E_2$ are systematic. The containment $\supseteq$ follows from the definition, since every $y \otimes z$ for $y \in C_1$, $z \in C_2$ is a codeword. Let us prove the containment $\subseteq$: For every $i \in [k_2]$, $j \in [k_1]$, let $\Delta_{i,j}$ be the $k_2 \times k_1$ matrix whose $(i,j)$ entry is 1, and all its other entries are 0. Then, $\{(E_1 \otimes E_2)(\Delta_{i,j})\}_{i,j}$ forms a basis to $C_1 \otimes C_2$. Let $y = (y_1, \ldots, y_{n_1}) \in C_1$ be the $E_1$ encoding of the message which is 0 on all coordinates except for the $j$'th which is 1. Let $z = (z_1, \ldots, z_{n_2}) \in C_2$ be the $E_2$ encoding of the message which is 0 on all coordinates except for the $i$'th which is 1. The columns of $(E_1 \otimes E_2)(\Delta_{i,j})$ are percisely $y_1 z, \ldots, y_{n_1} z$. I.e., $(E_1 \otimes E_2)(\Delta_{i,j}) = y \otimes z$. $\square$